Unidade 3

Resolução

Objetivos:

- Conhecer alguns dos principais algoritmos que podem ser utilizados durante o processo de prova automática de argumentos e de programação em Lógica de Predicados.
- Aplicar o algoritmo de prova automática da Resolução e seus algoritmos auxiliares na obtenção de respostas para consultas realizadas a programas em Lógica de Predicados.

Capítulo 1

Conversão para Forma Clausal

A Resolução é um método de decisão que, em uma única operação, executa a série de rocessos envolvidos com declarações em Linguagem Lógica de Predicados.

Esse método opera em declarações que foram convertidas em uma forma padrão conveniente: a **Forma Clausal**.

As Respostas são obtidas por meio de um **processo de prova por refutação**.

Definições:

(a) Uma fórmula P é uma **Conjunção** se e somente se, omitindo-se os parênteses, for da forma $P_1 \wedge P_2 \wedge ... \wedge P_n$.

(b) Uma fórmula P é uma **Disjunção** se e somente se, omitindo-se os parênteses, for da forma $P_1 \vee P_2 \vee ... \vee P_n$.

(c) Uma fórmula P está em **Forma Normal Prenex** se e somente se P for da forma Q(M) onde Q, o prefixo de P, é uma cadeia de quantificadores e M, a matriz de P, é uma fórmula sem ocorrência de quantificadores.

(d) Uma fórmula P está em **Forma Normal Conjuntiva** se e somente se estiver em Forma Normal Prenex e sua matriz for uma conjunção de disjunções de fórmulas atômicas, negadas ou não.

(e) Uma fórmula P está em **Forma Clausal** se e somente se estiver em Forma Normal Conjuntiva mas sem nenhuma instância do conectivo $\wedge$.

Exemplo:

- Fórmula em Forma Normal Prenex: $\forall x:(\exists y:(p(x,y) \rightarrow (q(x) \wedge q(y))))$

  Prefixo: $\forall x:\exists y:$

  Matriz: $p(x,y) \rightarrow (q(x) \wedge q(y))$
- Fórmula em Forma Normal Conjuntiva: $\forall x:(\exists y:((\neg p(x,y) \vee q(x)) \ (\neg p(x,y) \vee \ q(y))))$
- Fórmula na Forma Clausal

  Cláusula1: $\neg p(x,F(x)) \vee q(x)$

  Cláusula2: $\neg p(x,F(x)) \vee q(F(x))$

Algoritmo Conversão para forma Clausal

Para que a resolução funcione, primeiro, precisamos converte cada fórmula do programa em Forma Normal Conjuntiva e, em seguida, dividir cada expressão resultante em um conjunto de cláusulas.

A partir do algoritmo de conversão, considerando uma fórmula em Linguagem Lógica de Predicados e a execução dos passos abaixo sobre a fórmula, obteremos um conjunto de cláusulas equivalente a fórmula original.

1. Elimine $\rightarrow$, usando:

$(p \rightarrow q) \leftrightarrow (\neg p \vee q)$. - Tautologia 12.a).

2. Reduza o escopo de cada $\neg$ a um único termo usando:

$\neg\neg p \leftrightarrow p$, - Tautologia 5,

$\neg(p \wedge q) \leftrightarrow (\neg p \vee \neg q)$ e - Tautologia 10.a),

$\neg(p \wedge q) \leftrightarrow (\neg p \vee \neg q)$. - Tautologia 10.b);

e as correspondências padrões entre quantificadores:

$\neg(\forall x{:}p(x)) \leftrightarrow \exists x{:}\neg p(x)$ e

$\neg(\exists x{:}p(x)) \leftrightarrow \forall x{:}\neg p(x)$.

3. Padronize as variáveis para que cada quantificador fique vinculado a uma única variável, por exemplo:

$(\forall x{:}p(x)) \vee (\forall x{:}q(x))$

poderia ser convertida em

$(\forall x{:}p(x)) \vee (\forall y{:}q(y))$.

4. Coloque a fórmula na Forma Normal Prenex.

5. Elimine os quantificadores existenciais, ou seja:

$\exists y{:}presidente(y)$

pode ser transformada em $presidente(S_1)$,

e

$\forall x{:}\exists y{:}pai\_de(y,x)$

pode ser transformada em $\forall x{:}pai\_de(S_2(x),x)$

onde:

S_1 - é uma função sem argumentos, que de algum modo produz um valor que satisfaz presidente;

S_2 - é uma função com um argumento, x, que de algum modo pro duz para cada x (em D) um valor que satisfaz pai_de.

6. Elimine o prefixo.

7. Converta a matriz em uma conjunção de disjunções, usando:

$(p \vee (q \wedge r)) \leftrightarrow ((p \vee q) \wedge (p \vee r))$. - Tautologia 8.b)

8. Crie uma cláusula separada que corresponda a cada conjunção:

c_1: $p \vee q$

e

c_2: $p \vee r$.

9. Padronize distintamente as variáveis do conjunto de cláusulas gera-dos no Passo 8. Esta transformação está baseada no fato que:

$(\forall x{:}(p(x) \wedge q(x))) \leftrightarrow ((\forall x{:}p(x)) \wedge (\forall x{:}q(x)))$.

Veja os exemplos a seguir:

Converter a fórmula abaixo para Forma Clausal

$\forall x:((Romano(x) \wedge conhece(x,Marcos)) \rightarrow (odeia(x,César) \vee (\forall y:((\exists z:odeia(y,z)) \rightarrow acha\_louco(x,y)))))$

1. $\forall x:(\neg(Romano(x) \wedge conhece(x,Marcos)) \vee odeia(x,César) \vee (\forall y:(\neg(\exists z:odeia(y,z)) \vee acha\_louco(x,y))))$

2. $\forall x:(\neg Romano(x) \vee \neg conhece(x,Marcos) \vee odeia(x,César) \vee (\forall y:((\forall z:\neg odeia(y,z)) \vee acha\_louco(x,y))))$

3. $\forall x:(Romano(x) \vee \neg conhece(x,Marcos) \vee odeia(x,César) \vee (\forall y:((\forall z:\neg odeia(y,z)) \vee acha\_louco(x,y))))$

4. $\forall x:(y:(\forall z:(\neg Romano(x) \vee \neg conhece(x,Marcos) \vee odeia(x,César) \vee \neg odeia(y,z) \vee acha\_louco(x,y))))$

5. $\forall x:(y:(\forall z:(\neg Romano(x) \vee \neg conhece(x,Marcos) \vee odeia(x,César) \vee \neg odeia(y,z) \vee acha\_louco(x,y))))$

6. $\neg Romano(x) \vee \neg conhece(x,Marcos) \vee odeia(x,César) \vee \neg odeia(y,z) \vee acha\_louco(x,y)$

7. $\neg Romano(x) \vee \neg conhece(x,Marcos) \vee odeia(x,César) \vee \neg odeia(y,z) \vee acha\_louco(x,y)$

8. $\neg Romano(x) \vee \neg conhece(x,Marcos) \vee odeia(x,César) \vee \neg odeia(y,z) \vee acha\_louco(x,y)$

9. $\neg Romano(x) \vee \neg conhece(x,Marcos) \vee odeia(x,César) \vee \neg odeia(y,z) \vee acha\_louco(x,y)$

Aplicação do algoritmo ao programa que nos fala sobre o mundo de Marcos e César

1. $homem(Marcos)$
homem(Marcos)

2. $Pompeano(Marcos)$
Pompeano(Marcos)

3. $\forall x:(Pompeano(x) \rightarrow Romano(x))$
3.1 $\forall x:(\neg Pompeano(x) \vee Romano(x))$
3.6 $\neg Pompeano(x) \vee Romano(x)$
3.9 $\neg Pompeano(x_1) \vee Romano(x_1)$

4. $soberano(César)$
soberano(César)

5. $\forall x:(Romano(x) \rightarrow (leal\_a(x,César) \vee odeia(x,César)))$
5.1 $\forall x:(\neg Romano(x) \vee leal\_a(x,César) \vee odeia(x,César))$
5.6 $\neg Romano(x) \vee leal\_a(x,César) \vee odeia(x,César)$
5.9 $\neg Romano(x_2) \vee leal\_a(x_2,César) \vee odeia(x_2,César)$

6. $tentou\_assassinar(Marcos,César)$
tentou_assassinar(Marcos,César)

7. $\forall x:(\forall y:((pessoa(x) \wedge soberano(y) \wedge tentou\_assassinar(x,y)) \rightarrow \neg leal\_a(x,César)))$
7.1 $\forall x:(\forall y:(\neg(pessoa(x) \wedge soberano(y) \wedge tentou\_assassinar(x,y)) \vee \neg leal\_a(x,César)))$

7.2 $\forall x:(\forall\ y:(\neg pessoa(x) \vee \neg soberano(y) \vee \neg tentou\_assassinar(x,y) \vee \neg leal\_a(x,César)))$

7.6 $\neg pessoa(x) \vee \neg soberano(y) \vee \neg tentou\_assassinar(x,y) \vee \neg leal\_a(x,César)$

7.9 $\neg pessoa(x_3) \vee \neg soberano(y) \vee \neg tentou\_assassinar(x_3,y) \vee \neg leal\_a(x_3,César)$

8. $\forall x:(\exists y:leal\_a(x,y))$

8.5 $\forall x:(leal\_a(x,S_1(x)))$

8.6 $leal\_a(x,S_1(x))$

8.9 $leal\_a(x_4,S_1(x_4))$

9. $\forall x:(homem(x) \rightarrow pessoa(x))$

9.1 $\forall x:(\neg homem(x) \vee pessoa(x))$

9.6 $\neg homem(x) \vee pessoa(x)$

9.9 $\neg homem(x_5) \vee pessoa(x_5)$

Novo programa:

Cláusula1: $homem(Marcos)$

Cláusula2: $Pompeano(Marcos)$

Cláusula3: $\neg Pompeano(x_1) \vee Romano(x_1)$

Cláusula4: $soberano(César)$

Cláusula5: $\neg Romano(x_2) \vee leal\_a(x_2,César) \vee odeia(x_2,César)$

Cláusula6: $tentou\_assassinar(Marcos,César)$

Cláusula7: $\neg pessoa(x_3) \vee \neg soberano(y) \vee \neg tentou\_assassinar(x_3,y) \vee \neg leal\_a(x_3,César)$

Cláusula8: $leal\_a(x_4,S_1(x_4))$

Cláusula9: $\neg homem(x_5) \vee pessoa(x_5)$

ATIVIDADES DE AVALIAÇÃO

1. Colocar em forma normal conjuntiva a seguinte fórmula:

$\forall x:(cidade(x) \rightarrow (\exists y:guarda(x,y) \wedge (\forall z:((cão(z) \wedge vive(x,z)) \rightarrow mordido(y,z)))))$.

Em seguida, dar três passos adicionais que completarão o processo de conversão da fórmula original em uma conjunção de cláusulas, útil no método da resolução:

- Eliminar quantificadores universais,
- Eliminar símbolos de conjunção e
- Rebatizar variáveis de forma que um certo termo do tipo variável não apareça em mais que uma cláusula.

2. Converta para forma clausal as seguintes fórmulas:

a) $\forall x:(\forall y:(P(x,y) \rightarrow Q(x,y)))$.

b) $\forall x:(\forall y:(\neg Q(x,y) \rightarrow \neg P(x,y)))$.

c) $\forall x:(\forall y:(P(x,y) \rightarrow (Q(x,y) \rightarrow R(x,y))))$.

d) $\forall x:(\forall y:(P(x,y) \wedge (Q(x,y) \rightarrow R(x,y))))$.

e) $\forall x:(\forall y:(P(x,y) \rightarrow (Q(x,y) \vee R(x,y))))$.

f) $\forall x:(\forall y:(P(x,y) \rightarrow (Q(x,y) \wedge R(x,y))))$.

g) $\forall x:(\forall y:((P(x,y) \vee Q(x,y))) \rightarrow R(x,y)))$.

h) $\forall x:(\exists y:(P(x,y) \rightarrow Q(x,y)))$.

i) $\neg(\forall x:(\exists y:(P(x,y) \rightarrow Q(x,y))))$.

j) $\neg(\forall x:P(x)) \rightarrow (\exists x:P(x))$.

k) $\forall x:(P(x) \rightarrow P(x))$.

l) $\neg(\forall x:P(x)) \rightarrow (\exists x:\neg P(x))$.

Capítulo 2

Algoritmo da Unificação

Na Resolução para Lógica Proposicional é fácil para um programa provador automático obter uma cláusula-resolvente a partir de duas cláusulas-pais, contendo símbolos proposicionais complementares causadores de uma contradição. Por exemplo:

$$\underbrace{\neg \boldsymbol{p} \vee q \vee r \qquad\qquad \boldsymbol{p} \vee s}$$
$$q \vee r \vee s.$$

No exemplo, o programa procurou uma contradição, obteve a cláusula-resolvente como uma disjunção das cláusulas-pais e, em seguida, eliminou a contradição, eliminando os símbolos complementares $\neg p$ e p. Esta facilidade decorre de um processo de casamento simples, proporcionado pela presença de simples **símbolos proposicionais**.

Na Resolução para a Lógica de Predicados o processo de casamento entre **fórmulas atômicas** torna-se mais complicado, pois os argumentos dos predicados que compõem estas fórmulas também devem ser comparados. Neste caso, podem ocorrer diversos tipos de situações. Por exemplo:

- Situação em que **se configura uma contradição** sem substituição de variáveis:

$$\underbrace{\neg \boldsymbol{homem(João)} \vee mulher(Maria) \qquad\qquad \boldsymbol{homem(João)} \vee mortal(x)}$$
$$\{\ \}$$
$$mulher(Maria) \vee mortal(x)$$

- Situação em que **se configura uma contradição** com substituição de variáveis:

$$\underbrace{\neg \boldsymbol{homem(João)} \vee mulher(Maria) \qquad\qquad \boldsymbol{homem(x)} \vee mortal(x)}$$
$$\{\ \boldsymbol{(João/x)}\ \}$$
$$mulher(Maria) \vee mortal(João)$$

- Situação em que **não se configura uma contradição:**

$$\underbrace{\neg \boldsymbol{homem(João)} \vee mulher(Maria) \qquad\qquad \boldsymbol{homem(Pedro)} \vee mortal(x)}$$
FALHA

Para auxiliar o programa provador automático na busca de contradições, podemos implementar certo algoritmo denominado **Algoritmo da Unificação**. Este algoritmo descreve uma função recursiva que compara duas fórmulas atômicas e retorna, se existir, um conjunto de substituições que tornem as fórmulas idênticas.

O Algoritmo

Unificar(*L1, L2*)

1. Se *L1* ou *L2* é variável ou constante

então se *L1* e *L2* são idênticos

então **retorne *{ }*.**

senão se *L1* for variável

então se *L1* ocorre em *L2*

então **retorne *FALHA*.**

senão **retorne *{ (L2/L1) }*.**

senão se *L2* for variável

então se *L2* ocorre em *L1*

então **retorne *FALHA*.**

senão **retorne *{ (L1/L2) }*.**

senão **retorne *FALHA*.**

2. Se os símbolos predicativos em *L1* e *L2* não são idênticos

então **retorne *FALHA*.**

3. Se *L1* e *L2* têm número de termos diferentes

então **retorne *FALHA*.**

4. Faça *SUBST* := *{ }*.

5. Para i = 1, ..., número de termos em *L1*:

(a) Chame Unificar com o i-ésimo argumento de *L1* e o i-ésimo argumento de *L2*, colocando o resultado em *S*.

(b) Se *S* = *FALHA*

então **retorne *FALHA*.**

(c) Se *S* ≠ *{ }*

então: aplique *S* ao restante de *L1* e *L2* e

faça *SUBST* := Concatena(*SUBST, S*).

6. Retorne *SUBST*..

Observação sobre a função Concatena(*Lista1, Lista2*):

A função Concatena recebe como argumentos duas listas de substituições, armazenadas em *Lista1* e *Lista2*, e retorna uma terceira lista resultante da concatenação das duas primeiras. Por exemplo:

SUBST := Concatena(*{ (Marcos/x) }, { (z/y) }*)
SUBST := *{ (Marcos/x), (z/y) }*

SUBST := Concatena(*{ (z/y) }, { (Marcos/x) }*)
SUBST := *{ (z/y), (Marcos/x) }*

SUBST := Concatena(*{ }, { (Marcos/x), (z/y) }*)
SUBST := *{ (Marcos/x), (z/y) }*

SUBST := Concatena({ *(Marcos/x), (z/y)* }, { })
SUBST := { *(Marcos/x), (z/y)* }

Exemplos:

Unificar(*homem(João), homem(João)*)

L1 := homem(João)
L2 := homem(João)

4. *SUBST* := { } .
5. i := 1
(a) Unificar(João, João)
S := { }.
6. Retornar { }.

Unificar(*homem(João), homem(x)*)

L1 := homem(João)
L2 := homem(x)

4. *SUBST* := { }.
5. i := 1
(a) Unificar**(***João, x***)**
S := { (João/x) }.
(c) *L1 := homem(João)*
L2 := homem(x)
SUBST := { (João/x) }.
6. Retornar *{ (João/x) }.*

Unificar(*homem(João), homem(Pedro)*)

L1 := homem(João)
L2 := homem(Pedro)

4. *SUBST* := { }.
5. i := 1
(a) Unificar(João, Pedro)
S := FALHA.
(b) retornar FALHA.

Unificar(*odeia(Marcos,z), odeia(x,y)*)

L1 := odeia(Marcos, z)
L2 := odeia(x,y)

4. *SUBST := { }.*
5. i := 1
(a) Unificar(Marcos, x)
S := { (Marcos/x)}
(c) *L1 := odeia(Marcos,z)*
L2 := odeia(x,y)
SUBST := { (Marcos/x) }

i := 2

(a) Unificar(z, y)

S := { (y/z) }.

(c) *L1 := odeia(Marcos,z)*

L2 = odeia(x,y)

SUBST := { (Marcos/x), (y/z) }.

6. Retornar { (y/z), (Marcos/x) }.

Unificar(*f(x,x), f(g(x),x)*)

L1 := f(x,x)

L2 := f(g(x),x)

4. *SUBST := { }.*

5. i := 1

(a) Unificar(x, g(x))

S := FALHA.

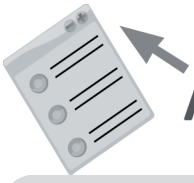
(b) retornar FALHA.

Observação:

Este último exemplo descreve um tipo de situação em que o Algoritmo da Unificação tenta unificar uma expressão funcional *g(x)*, envolvendo a variável *x*, com a própria variável *x*. Conforme se pode observar, neste caso, o Algoritmo *retorna* ***FALHA***, pois se ele aceitasse *g(x)* como substituição para *x*, *S := { (g(x)/x) }*, então quando ele fosse substituir *x* por *g(x)* no restante da fórmula atômica *L1* (Passo 5.(c)), ele estaria causando uma recursão infinita, ou seja:

L1 := f(x,g(g(g(g... ;

já que não seria possível eliminar *x de L1*.



ATIVIDADES DE AVALIAÇÃO

1. Utilize o Algoritmo da Unificação para produzir substituições apropriadas para os pares de fórmulas atômicas abaixo. Caso possível, ou seja, caso as fórmulas sejam unificáveis, expresse o retorno do Algoritmo. Caso não possível, ou seja, caso as fórmulas não sejam unificáveis, explique porque.

a) *cor(lolo,amarelo)*	*cor(x,y)*
b) *cor(lolo,amarelo)*	*cor(x,x)*
c) *cor(chapéu(carteiro),azul)*	*cor(chapéu(y),x)*
d) *R(F(x),bbb)*	*R(y,z)*
e) *R(F(y),x)*	*R(x,F(bbb))*
f) *R(F(y),y,x)*	*R(x,F(z),F(w))*
g) *ama(x,y)*	*ama(y,x)*

Capítulo 3

Algoritmo da Resolução

Basicamente, os passos que compõem o **algoritmo da Resolução para a Lógica de Predicados** são os mesmos realizados para a Lógica Proposicional. As diferenças estão nos algoritmos de Conversão para Forma Clausal e, no caso da Lógica de Predicados, a necessidade de se ter em mãos um procedimento de unificação, que possibilite verificar o casamento de duas fórmulas atômicas.

Seja: ***P*** = ***Programa*** = *{ Fórmulas em Lógica de Predicados } e*
C = ***Consulta*** *a ser respondida pelo Algoritmo da Resolução = Fórmula em Lógica de Predicados*

1. Converter Fórmulas de ***P*** para Forma Clausal.

2. Negar ***C***. Converter ¬***C*** para Forma Clausal e acrescentar resultado ao novo ***P***, obtido no Passo 1.

3. Repetir {

a) Selecionar duas ***cláusulas-pais, C1*** e ***C2***, tal que:

C1 *contém fórmula atômica* ***f***,

C2 contém *fórmula* ¬***f'***, e

f *unifica com* ***f'***;

b) Obter:

resolvente = *disjunção de todos as fórmulas atômicas de* ***C1*** e ***C2*** com a eliminação de ***f*** e ¬***f'***, realizando-se as substituições apropriadas na disjunção resultante*;*

c) Se ***resolvente*** ≠ **nil**

então acrescentar ***resolvente*** a ***P***.

senão ***Resposta*** foi encontrada.

} até uma ***Resposta*** ser encontrada, ou até não ser possível nem um progresso, ou até uma quantidade pré-determinada de repetições terem sido realizadas.

4. Responder.

Conforme você pode perceber, o Algoritmo da Resolução para a Lógica de Predicados, primeiramente, chama várias vezes o algoritmo de Conversão para a Forma Clausal no Passo 1 e uma vez no Passo 2; posteriormente, chama, sempre que necessário, o algoritmo da Unificação em cada repetição no Passo 3.

Veja outros exemplos:

$P = \{F_1.\ homem(Marcos)$,
$F_2.\ Pompeano(Marcos)$,
$F_3.\ \forall x:(Pompeano(x) \rightarrow Romano(x))$,
$F_4.\ soberano(César)$,
$F_5.\ \forall x:(Romano(x) \rightarrow (leal\_a(x,César) \vee odeia(x,César)))$,
$F_6.\ tentou\_assassinar(Marcos,César)$,
$F_7.\ \forall x:(\forall y:((pessoa(x) \wedge soberano(y) \wedge tentou\_assassinar(x,y)) \rightarrow \neg leal\_a(x,y)))$,
$F_8.\ \forall x:(\exists y:leal\_a(x,y))$,
$F_9.\ \forall x:(homem(x) \rightarrow pessoa(x))$
$\}$
$C = odeia(Marcos,César)$

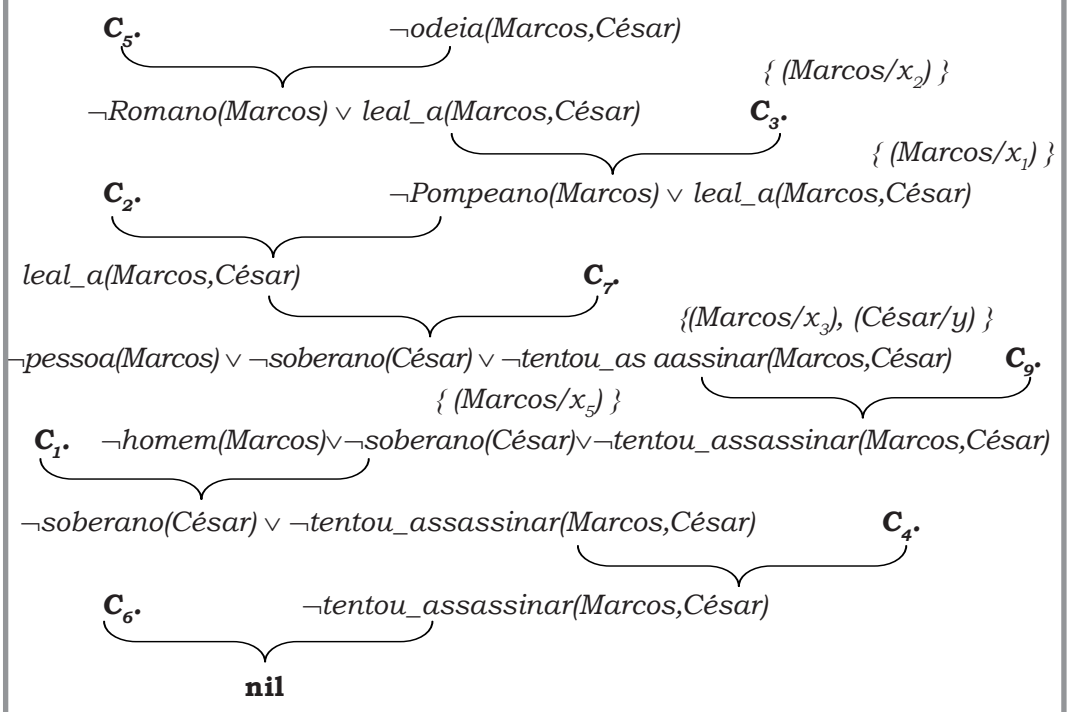
1.

$P = \{ C_1.\ homem(Marcos)$,
$C_2.\ Pompeano(Marcos)$,
$C_3.\ \neg Pompeano(x_1) \vee Romano(x_1)$,
$C_4.\ soberano(César)$,
$C_5.\ \neg Romano(x_2)\ \ leal\_a(x_2,César) \vee odeia(x_2,César)$,
$C_6.\ tentou\_assassinar(Marcos,César)$,
$C_7.\ \neg pessoa(x_3) \vee \neg soberano(y) \vee \neg tentou\_assassinar(x_3,y) \vee \neg leal\_a(x_3,y)$,
$C_8.\ leal\_a(x_4,S_1(x_4))$,
$C_9.\ \neg homem(x_5) \vee pessoa(x_5)$
$\}$.

2.

$\neg C = \neg odeia(Marcos,César)$.
$P = \{ C_1.\ homem(Marcos)$,
$C_2.\ Pompeano(Marcos)$,
$C_3.\ \neg Pompeano(x_1) \vee Romano(x_1)$,
$C_4.\ soberano(César)$,
$C_5.\ \neg Romano(x_2) \vee leal\_a(x_2,César) \vee odeia(x_2,César)$,
$C_6.\ tentou\_assassinar(Marcos,César)$,
$C_7.\ \neg pessoa(x_3) \vee \neg soberano(y) \vee \neg tentou\_assassinar(x_3,y) \vee \neg leal\_a(x_3,y)$,
$C_8.\ leal\_a(x_4,S_1(x_4))$,
$C_9.\ \neg homem(x_5) \vee pessoa(x_5)$,
$C_{10}.\ \neg odeia(Marcos,César)$
$\}$.

3.



4. *Resposta = 'sim'*.

Heurísticas que podem ser utilizadas para acelerar o processo de busca de uma resposta:

- Selecionar Cláusulas-pais que contenham fórmulas atômicas complementares.
- Eliminar resolventes que sejam:
  - tautologias,
  - subordinados a outras cláusulas existentes ($p \vee q$ está subordinado a p).
- Iniciar processo de resolução resolvendo a cláusula que se deseja refutar (consulta negada em forma clausal) e, sempre que possível, continuar resolvendo os resolventes gerados a partir desta.
- Sempre que possível, resolver cláusulas-pais que contenham uma única fórmula atômica.

Situações nas quais a Resolução pode detectar que não existe contradição:

- No programa não existe uma cláusula que contenha uma fórmula atômica complementar a alguma fórmula atômica da consulta negada em forma clausal, por exemplo:

P = { C_1. *homem(Marcos)* ,
C_2. *Pompeano(Marcos)* ,
C_3. $\neg$*Pompeano*(x_1) $\vee$ *Romano*(x_1) ,
C_4. *soberano(César)* ,
C_5. $\neg$*Romano*(x_2) *leal_a*(x_2,*César*) $\vee$ *odeia*(x_2,*César*) ,

Continuação...

C_6. *tentou_assassinar(Marcos,César) ,*

C_7. $\neg$*pessoa(x_3)*$\vee\neg$*soberano(y)* $\vee$ $\neg$*tentou_assassinar(x_3,y)*$\vee$ $\neg$*leal_a(x_3,y),*

C_8. *leal_a(x_4,S_1(x_4)) ,*

C_9. $\neg$*homem(x_5)* $\vee$ *pessoa(x_5)*

}.

$\neg C$ = *odeia(Marcos,César).*

- Quando a Resolução partir de uma certa consulta negada e gerar um tipo de situação semelhante a situação descrita acima, por exemplo:

$\neg C$ = $\neg$*leal_a(Marcos,César).*

Resolução:

C_5. $\neg$*leal_a(Marcos,César).*

{ (Marcos/x_2) }

$\neg$*Romano(Marcos)* $\vee$ *odeia(Marcos,César)* C_3.

{ (Marcos/x_1) }

C_2. $\neg$*Pompeano(Marcos)* $\vee$ *odeia(Marcos,César)*

odeia(Marcos,César) **?**

Resolução lidando com Funções e Predicados Computáveis e a Noção de Igualdade:

P = {

F_1. *homem(Marcos) ,*

F_2. *Pompeano(Marcos) ,*

F_3. *nasceu(Marcos,40) ,*

F_4. $\forall$*x:(homem(x)* $\rightarrow$ *mortal(x)) ,*

F_5. *entrou_em_erupção(vulcão,79)* $\wedge$ $\forall$*x:(Pompeano(x)* $\rightarrow$ *morreu(x,79)) ,*

F_6. $\forall$*x:(*$\forall t_1$*:(*$\forall t_2$*:((mortal(x)* $\wedge$ *nasceu(x,t_1)* $\wedge$ **maq**(-(t_2,t_1),150))* $\rightarrow$ *morto(x,t_2))))*

F_7. *agora = 1998 ,*

F_8 $\forall$*x:(*$\forall$*t:(morto(x,t)* $\rightarrow$ $\neg$*vivo(x,t)) ,*

F_9. $\forall$*x:(*$\forall$*t:(*$\neg$*morto(x,t)* $\rightarrow$ *vivo(x,t)) ,*

F_{10}. $\forall$*x:(*$\forall t_1$*:(*$\forall t_2$*:((morreu(x,t_1)* $\wedge$ **maq**(t_2,t_1))* $\rightarrow$ *morto(x,t_2))))*

}

C = $\neg$*vivo(Marcos,agora)*

1.

$P = \{$ C_1. $homem(Marcos)$,
C_2. $Pompeano(Marcos)$,
C_3. $nasceu(Marcos,40)$,
C_4. $\neg homem(x_1) \vee mortal(x_1)$,
C_5. $\neg Pompeano(x_2)$ $morreu(x,79)$,
C_6. $entrou\_em\_erupção(vulcão,79)$,
C_7. $\neg mortal(x) \vee \neg\ nasceu(x,t_1) \vee \neg\ \mathbf{maq}(-(t_2,t_1),150) \vee\ morto(x,t_2)$
C_8. $agora = 1998$,
C_9. $\neg morto(x,t) \vee\ \neg vivo(x,t)$,
C_{10}. $morto(x,t) \vee vivo(x,t)$,
C_{11}. $\neg morreu(x,t_1) \vee \neg\mathbf{maq}(t_2,t_1) \vee\ morto(x,t_2)$
$\}$.

2.

$\neg C = vivo(Marcos,agora)$.

$P = \{$ C_1. $homem(Marcos)$,
C_2. $Pompeano(Marcos)$,
C_3. $nasceu(Marcos,40)$,
C_4. $\neg homem(x_1) \vee mortal(x_1)$,
C_5. $\neg Pompeano(x_2)$ $morreu(x_2,79)$,
C_6. $entrou\_em\_erupção(vulcão,79)$,
C_7. $\neg mortal(x_3) \vee \neg\ nasceu(x_3,t_1) \vee \neg\ \mathbf{maq}(-(t_2,t_1),150) \vee\ morto(x_3,t_2)$,
C_8. $agora = 1998$,
C_9. $\neg morto(x_4,t_3) \vee\ \neg vivo(x_4,t_3)$,
C_{10}. $morto(x_5,t_4) \vee vivo(x_5,t_4)$,
C_{11}. $\neg morreu(x_6,t_5) \vee \neg\mathbf{maq}(t_6,t_5) \vee\ morto(x_6,t_6)$
C_{12}. $vivo(Marcos,agora)$
$\}$

3.

C_9. $vivo(Marcos,agora)$
$\{ (Marcos/x_4), (agora/t_3) \}$
$\neg morto(Marcos,agora)$ C_{11}.
$\{ (Marcos/x_6), (agora/t_6) \}$
C_5. $\neg morreu(Marcos,t_5) \vee \neg\mathbf{maq}(agora,t_5)$
$\{ (Marcos/x_2), (79/t_5) \}$
$\neg Pompeano(Marcos) \vee \neg\mathbf{maq}(agora,79)$ C_8.
substituir iguais
$\neg Pompeano(Marcos) \vee \neg\mathbf{maq}(1998,79)$
Computar ***maq***
C_2. $\neg Pompeano(Marcos)$
nil

4. *Resposta = 'sim'*.

Formas básicas em que as **consultas** em Linguagem Natural podem aparecer e tipos de **respostas** correspondentes:

- Perguntas que exigem respostas do **tipo *sim-não***:
  - Consulta em Linguagem Natural: *"Marcos está vivo ?"*
  - Consulta em Lógica de Predicados: $\neg morto(Marcos,agora)$
  - Respostas: *'sim"* ou *'não'*.
- Perguntas que exigem respostas do **tipo *preenchimento de lacunas***:
  - Consulta em Linguagem Natural: *"Quem tentou assassinar um soberano ?"*
  - Consulta em Lógica de Predicados: $\exists x:(\exists y:(\ tentou\_assassinar(x,y) \wedge soberano(y)))$
  - Respostas: *substituição de indivíduos do universo de discurso no lugar de x, digamos (x_0/x), e de y, digamos (y_0/y), tal que tentou_assassinar$(x_0,y_0) \wedge soberano(y_0)$ é verdadeiro.*

Exemplos:

$\boldsymbol{P}$ = {
$\boldsymbol{F_1}$. $homem(Marcos)$,
$\boldsymbol{F_2}$. $Pompeano(Marcos)$,
$\boldsymbol{F_3}$. $nasceu(Marcos,40)$,
$\boldsymbol{F_4}$. $\forall x:(homem(x) \rightarrow mortal(x))$,
$\boldsymbol{F_5}$. $entrou\_em\_erupção(vulcão,79) \wedge \forall x:(Pompeano(x) \rightarrow morreu(x,79))$,
$\boldsymbol{F_6}$. $\forall x:(\forall t_1:(\forall t_2:((\ mortal(x) \wedge nasceu(x,t_1) \wedge \boldsymbol{maq}(-(\ t_2,t_1),150)) \rightarrow morto(x,t_2))))$,
$\boldsymbol{F_7}$. $agora = 1998$,
$\boldsymbol{F_8}$ $\forall x:(\forall t:(morto(x,t) \rightarrow \neg vivo(x,t))$,
$\boldsymbol{F_9}$. $\forall x:(\forall t:(\neg morto(x,t) \rightarrow vivo(x,t))$,
$\boldsymbol{F_{10}}$. $\forall x:(\forall t_1:(\forall t_2:((\ morreu(x,t_1) \wedge \boldsymbol{maq}(\ t_2,t_1)) \rightarrow morto(x,t_2)))))$}
$\boldsymbol{C}$ = $\exists t:morreu(Marcos,t)$

1.
$\boldsymbol{P}$ = {
$\boldsymbol{C_1}$. $homem(Marcos)$,
$\boldsymbol{C_2}$. $Pompeano(Marcos)$,
$\boldsymbol{C_3}$. $nasceu(Marcos,40)$,
$\boldsymbol{C_4}$. $\neg homem(x_1) \vee mortal(x_1)$,
$\boldsymbol{C_5}$. $\neg Pompeano(x_2)$ $morreu(x,79)$,
$\boldsymbol{C_6}$. $entrou\_em\_erupção(vulcão,79)$,
$\boldsymbol{C_7}$. $\neg mortal(x) \vee \neg\ nasceu(x,t_1) \vee \neg\ \boldsymbol{maq}(-(\ t_2,t_1),150) \vee$ $morto(x,t_2)$,
$\boldsymbol{C_8}$. $agora = 1998$,
$\boldsymbol{C_9}$. $\neg morto(x,t) \vee \neg vivo(x,t)$,
$\boldsymbol{C_{10}}$. $morto(x,t) \vee vivo(x,t)$,
$\boldsymbol{C_{11}}$. $\neg morreu(x,t_1) \vee \neg\boldsymbol{maq}(\ t_2,t_1)) \vee$ $morto(x,t_2)$
}.

2.

$\neg C = \forall t{:}\neg morreu(Marcos,t).$

Convertendo-se a consulta negada para forma clausal obtemos:

$C_{12}. = \neg morreu(Marcos,t).$

$P = \{$ $C_1.$ $homem(Marcos)$,
$C_2.$ $Pompeano(Marcos)$,
$C_3.$ $nasceu(Marcos,40)$,
$C_4.$ $\neg homem(x_1) \vee mortal(x_1)$,
$C_5.$ $\neg Pompeano(x_2)$ $morreu(x_2,79)$,
$C_6.$ $entrou\_em\_erupção(vulcão,79)$,
$C_7.$ $\neg mortal(x_3) \vee \neg nasceu(x_3,t_1) \vee \neg$ **maq**$(-(t_2,t_1),150) \vee morto(x_3,t_2)$,
$C_8.$ $agora = 1998$,
$C_9.$ $\neg morto(x_4,t_3) \vee \neg vivo(x_4,t_3)$,
$C_{10}.$ $morto(x_5,t_4) \vee vivo(x_5,t_4)$,
$C_{11}.$ $\neg morreu(x_6,t_5) \vee \neg$**maq**$(t_6,t_5) \vee morto(x_6,t_6)$
$C_{12}.$ $\neg morreu(Marcos,t) \}$.

3.

$C_5.$ $\neg morreu(Marcos,t)$ $\{ (Marcos/x_2), (79/t) \}$

$Pompeano(Marcos)$ $C_2.$

nil

4. *Resposta = 79*.

Observações:

- A resposta pode ser derivada da cadeia de unificações que leva de volta à cláusula inicial.
- A resposta pode ser derivada acrescentando uma fórmula fictícia no processo de resolução. Esta fórmula consiste da consulta negada em forma clausal com um marcador especial. Por exemplo:

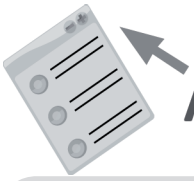
$C_5.$ $\neg morreu(Marcos,t) \vee \underline{\neg \mathbf{morreu(Marcos,t)}}$

$\{ (Marcos/x_2), (79/t) \}$

$Pompeano(Marcos) \vee \underline{\neg \mathbf{morreu(Marcos,79)}}$ $C_2.$

$\underline{\neg \mathbf{morreu(Marcos,79)}}$.

Neste caso, a Resolução pode parar quando só restar a fórmula fictícia.



ATIVIDADES DE AVALIAÇÃO

1. Considerando o par de Cláusulas-pais abaixo:

$C_1 = \neg P(z_1,a) \vee \neg P(z_1,x) \vee \neg P(x,z_1)$ e

$C_2 = P(z_2,f(z_2)) \vee P(z_2,a)$,

onde o símbolo "a" denota uma constante, encontre uma substituição adequada para os dois conjuntos de fórmulas atômicas componentes (omitindo-se a negação em C_1) e, logo após, encontre o resolvente correspondente.

2. Em relação ao método de RESOLUÇÃO, explique sucintamente porque é importante dispor de um processo de conversão de uma fórmula qualquer para uma fórmula equivalente expressa em forma normal conjuntiva.

3. Porque CASAMENTO é importante no método da RESOLUÇÃO?

4. Mostre que a Resolução é consistente; isto é, mostre que o resolvente de duas cláusulas-pais segue logicamente destas cláusulas.

5. Considere a seguinte base de conhecimento:

$\forall x:(y:(H(x) \wedge D(y)) \rightarrow F(x,y))$,

$\exists y:(G(y) \wedge (\forall z:(R(z) \rightarrow F(y,z))))$,

$\forall y:(G(y) \rightarrow D(y))$,

$\forall x:(\forall y:(\forall z:((F(x,y) \wedge F(y,z)) \rightarrow F(x,z))))$.

Use Resolução para responder a seguinte consulta:

$\forall x:(\forall z:((H(x) \wedge R(z)) \rightarrow F(x,z)))$.

6. Usando o primeiro programa em lógica desta Seção, responda à pergunta: *Marcos odiava César?"*

7. Anteriormente, mostramos que, dado um conjunto de fatos, havia duas maneiras de provar a declaração $\neg vivo(Marcos,agora)$. Nesta Seção, apresentamos uma prova de Resolução que corresponde a um desses métodos. Use a Resolução para derivar a outra prova da declaração, através da outra cadeia de raciocínio.

8. Considere as seguintes sentenças:

"João gosta de todo tipo de comida"

"Maçãs são comida"

"Frango é comida"

"Qualquer coisa que alguém coma e que não cause sua morte é comida"

"Paulo come amendoim e ainda está vivo"

"Susana come tudo o que Paulo come"

a) Traduza essas sentenças em Lógica de Predicados.

b) Converta cada uma das fórmulas acima em forma clausal.

c) Responda a pergunta: *"João gosta de amendoim?"*

d) Use a resolução para responder: *"O que Susana come?"* , isto é, $\exists x:come(Susana,x)$.

9. Considere os seguintes fatos:

"Os membros do Clube de Tranca da Rua Elmo são João, Salete, Pau lo e Helena"

"João é casado com Salete"

"Paulo é irmão de Helena"

"A esposa ou marido de cada pessoa casada membro do clube tam bém está no Clube"

"A última reunião do Clube foi na casa do João"

a) Represente estes fatos em Lógica de Predicados.

b) A partir dos fatos informados, a maioria das pessoas seria capaz de responder às seguintes perguntas:

"A última reunião do Clube foi na casa de Salete?"

"Helena não é casada?"

Será que você consegue construir provas de resolução para demonstrar a verdade de cada uma destas declarações, dados os fatos listados anteriormente. Faça-o, se possível. Caso contrário, acrescente os fatos necessários e depois crie as provas

10. Assuma os seguintes fatos:

"Carlos gosta de cursos fáceis"

"O curso de ciências é difícil"

"Todos os cursos do departamento de prendas domésticas são fáceis"

"BK 301 é um curso de prendas domésticas"

Use a Resolução para responder à pergunta:

"De que curso Carlos gostaria?"

11. Nesta Seção respondemos à pergunta: *"Quando Marcos morreu?"*, usando a resolução para mostrar que houve um tempo em que Marcos morreu. Usando os mesmos fatos, e o fato adicional

$\forall x:(\forall t_1:(morto(x,t_1) \rightarrow (t_2:(maq(t_1,t_2) \wedge morreu(x,t_2)))))$,

existe um outro modo de mostrar que houve um tempo em que Marcos morreu.

a) Faça uma prova de resolução desta outra cadeia de raciocínio.

b) Que resposta esta prova dará à pergunta: *"Quando Marcos morreu?"*

12. Se um curso é fácil, alguns estudantes no curso são felizes. Se um curso tem exame, nenhum estudante no curso é feliz. Use resolução para mostrar que, se um curso tem exame, o curso não é fácil.

13. Qualquer coisa que pode ler é alfabetizada. Alguns golfinhos são inteligentes, mas nenhum golfinho é alfabetizado. Use resolução para mostrar que algumas coisas inteligentes não sabem ler.

14. Vitor foi assassinado, e Artur, Bernadete e Karlene são suspeitos. Artur disse que ele não fez isso. Ele disse que Bernadete era amiga da vítima mas que Karlene odiava a vítima. Bernadete disse que ela estava fora da cidade no dia do assassinato e, além disso, ela disse que nem conhecia o rapaz. Karlene disse que ela é inocente e que ela viu Artur e Berna-

dete com a vítima logo após o assassinato. Assumindo que todo mundo (exceto o assassino) está falando a verdade, use resolução para resolver o crime.

15. Sabemos que cavalos são mais rápidos do que cães e que há um cachorrão, chamado Fig, que é mais rápido que todos os coelhos. É conhecido que Centelha é um cavalo e Pernalonga é um coelho. Use resolução para mostrar que Centelha é mais rápido que Pernalonga.

16. Considere o seguinte programa:

"Animal que tem pena não é mamífero"

"Animal que têm pêlo é mamífero e não é ave"

"Animal que não é mamífero é ave"

"Animal que não tem pena têm pêlo"

"Sabiá é um animal que voa e não é mamífero"

"Pingüim é um animal que é ave e não voa"

"Vaca é um animal que é mamífero e não voa"

"Morcego é um animal que voa e não é ave"

Consulte o programa para identificar um animal que voa e não tem pena. Mostre a seqüência de prova usada.

DADOS DOS AUTORES

Gustavo Augusto Lima de Campos

Cursou graduação em Engenharia Elétrica na Universidade Federal do Pará (1987), mestrado em Engenharia Elétrica na Universidade Federal de Uberlândia (1990) e doutorado em Engenharia Elétrica na Universidade Estadual de Campinas (2003). Atualmente é professor adjunto da Universidade Estadual do Ceará. Tem experiência na área de Ciência da Computação, com ênfase em Inteligência Artificial, atuando principalmente nos seguintes temas: agentes inteligentes, lógica, redes neurais artificiais, sistemas fuzzy, busca e problemas de tomada de decisão.

Jerffeson Teixeira de Souza

Recebeu o título de Ph.D. em Ciência da Computação, em 2004, pela School of Information Technology and Engineering (SITE) da University of Ottawa, Canadá. Ele é Bacharel (1998) e Mestre (2000) em Ciência da Computação pela Universidade Federal do Ceará (UFC). Ele é atualmente professor adjunto da Universidade Estadual do Ceará (UECE). Seus interesses de pesquisa são: Otimização em Engenharia de Software, Documentação e Aplicação de Padrões de Software e Estudo de Técnicas e Aplicação de Algoritmos de Mineração de Dados.